Neo Party Hall Booking Application

Objective:

Neo Party Hall is an application to be built as a product that can help customers to choose a Party Hall and book a Hall for Party for a specific date.

Users of the System:

1. Super Admin
2. Admin
3. Customer

Functional Requirements:

- Build an application that customers can book Party Hall.
- The application should have signup, login, profile, dashboard page for user and login,signup,profile , dashboard , add bike page for the admin.
- This application should have a provision to maintain a database for customer information, Admin information, booking information and room information.
- Also, an integrated platform required for customers , admin and super admin.
- Administration module to include options for adding / modifying / removing the existing product(s) and customer management.

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- Filters for products like Low to High or showcasing halls based on the customer’s price range, specific date etc.
- Email integration for intimating new personalized offers to customers.
- Multi-factor authentication for the sign-in process
- Payment Gateway

Output/ Post Condition:

- Records Persisted in Success & Failure Collections
- Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance (during Festival days, National holidays etc) • eCommerce -< 3 Sec

	• Admin application < 2 Sec • Non Peak Load Performance • eCommerce < 2 Sec • Admin Application < 2 Sec
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover
Logging & Auditing	• The system should support logging(app/web/DB) & auditing at all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	React 16+ Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Prerequisites (Do's and Don'ts):

1. As soon as the project mode is opened and set up, navigate to the settings of the Visual Studio Code by clicking on **File -> preferences -> Settings or (Ctrl + ,)**.

   a. Click on the settings icon located as the first icon on the right-hand side to open the **settings.json** file.

   b. paste the following code inside it.

```
{ "settings": { "files.exclude": { "**/node_modules": true } } }
```

   c. Then proceed with running the react project.

Note: The above step has to be repeated each time, the project mode is opened up.

2. The React app should run in port 8081. Do not run the react app in the port: 3000 or port: 4200.

3. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.

2. The id provided should be defined strictly with the following name **data-test-id="<your id>"**. Example : **<input data-test-id="username" />.** The naming convention has to be followed for all the id's mentioned.

3.

4. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.

5. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
6. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.
2. Manual routing should be restricted by using Protected Routes from the react router package. For example, if the user enters as http://localhost:3000/signup or http://localhost:3000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

Admin Side:

Action	URL	Method	Response
Admin Login	/admin/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/admin/signup	POST-Sends Admin Model data	Admin added
Admin Dashboard	/admin/dashboard	GET	Return All the Rooms in the hotel
Admin Add Room	/admin/addRoom	POST-Sends Room Data	Room Added

Admin Edit Room	/admin/editRoom	POST-Sends Room Data	Room Edited
Admin Delete Room	/admin/deleteRoom	POST-Sends Room ID	Room Deleted
Admin Profile	/admin/profile	POST-Sends Admin ID	Return Admin Profile Details
Edit Profile	/admin/editProfile	GET-Sends Admin ID	Return Admin Profile Details
Edit Profile	/admin/editProfile	POST-Sends Admin Model data	Edits Admin Profile

User Side:

Action	URL	Method	Response
User Login	/user/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/user/signup	POST-Sends User Model data	User added
User Dashboard	/user/dashboard	GET	Return all the Hotels available
Displaying Hotel Rooms	/user/rooms	POST - Sends Hotel name and admin ID of that hotel	Return all the rooms of the hotel
Room Details	/user/roomDetails	POST-Sends Room ID	Return room details
Booked Rooms	/user/bookings	POST-Sends User ID	Return user bookings

Super Admin:

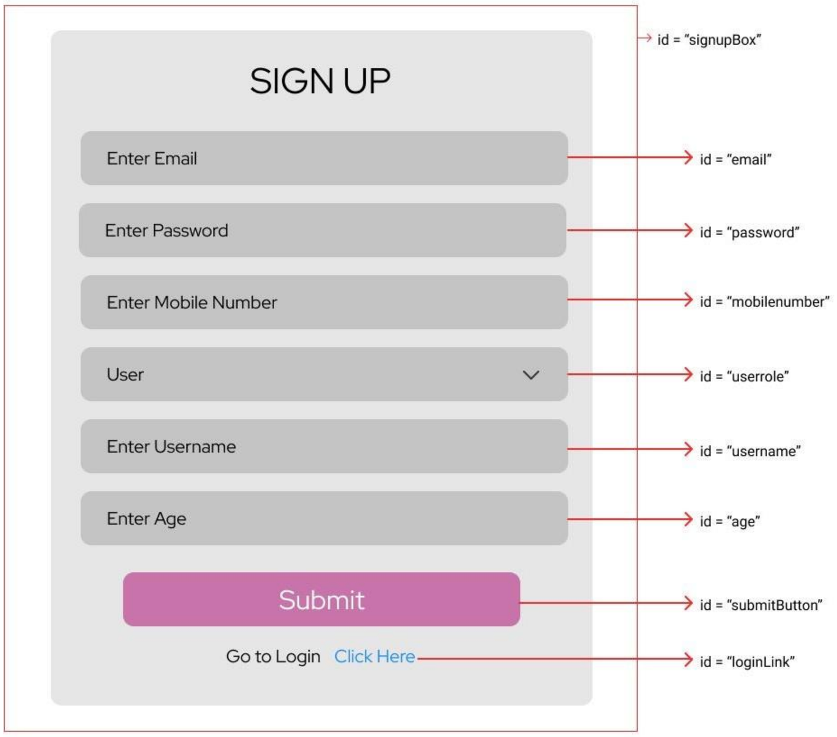
Action	URL	Method	Response

Super Admin Login	/super/login	POST-Sends email ID and password	Return True/False
Delete an admin	/super/deleteAdmin	POST-Sends admin email ID	Delete an admin
Delete an User	/super/deleteUser	POST-Sends user email ID	Delete a user

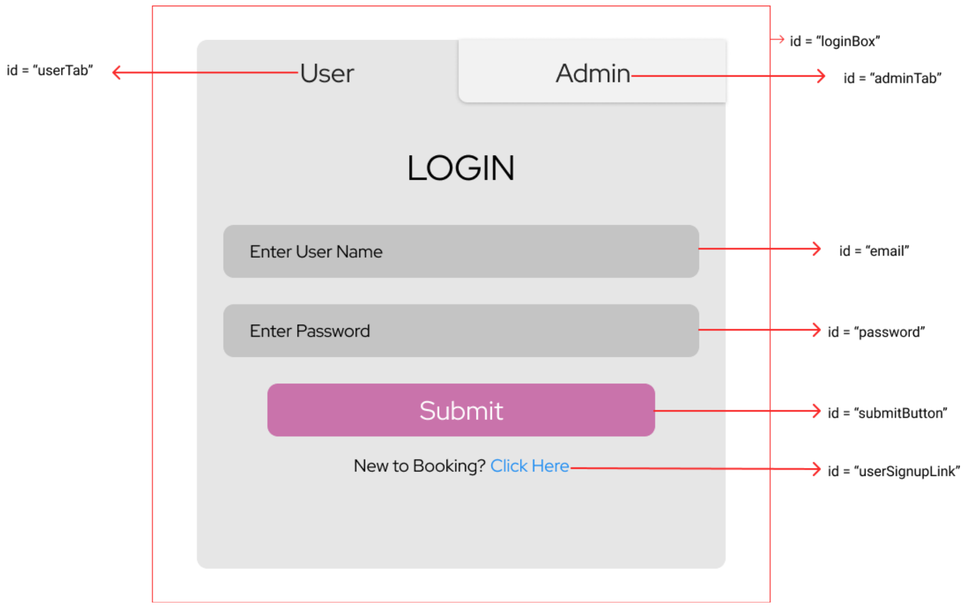
Frontend:

Customer:

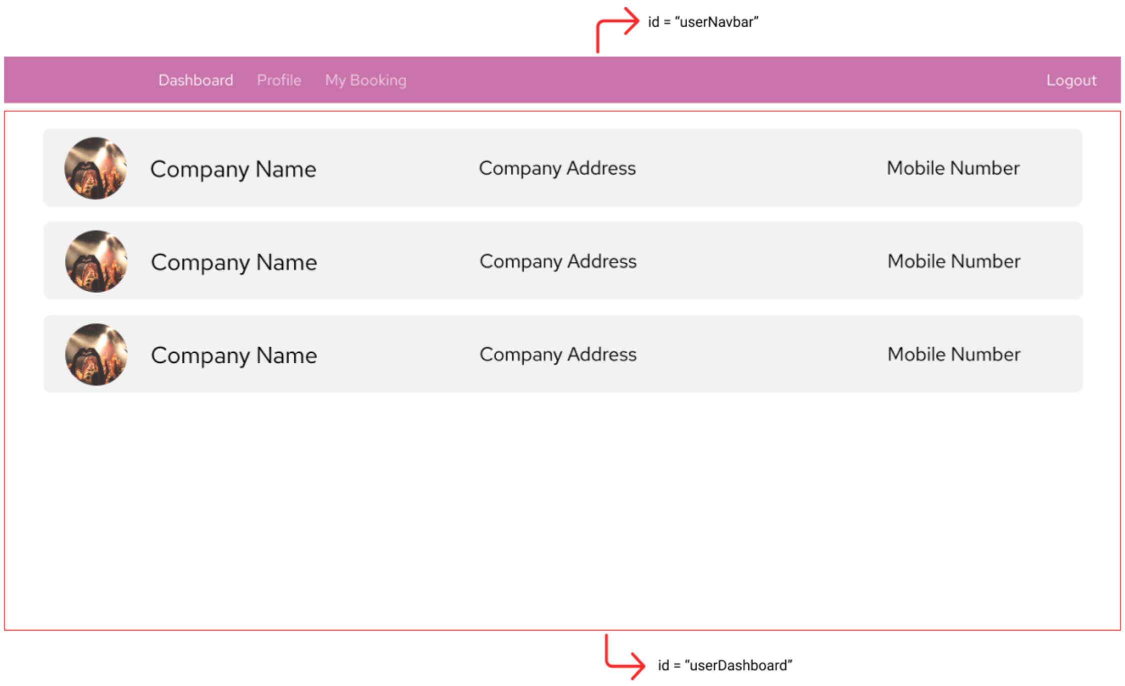
1. Signup: Design a signup page component where the new customer has options to sign up by providing their basic details.
   a. Ids:
      i. signupBox
      ii. email
      iii. password
      iv. mobilenumber
      v. userrole
      vi. username
      vii. age
      viii. submitButton
      ix. loginLink
   b. API endpoint Url: http://localhost:3000/signup
   c. Output screenshot:



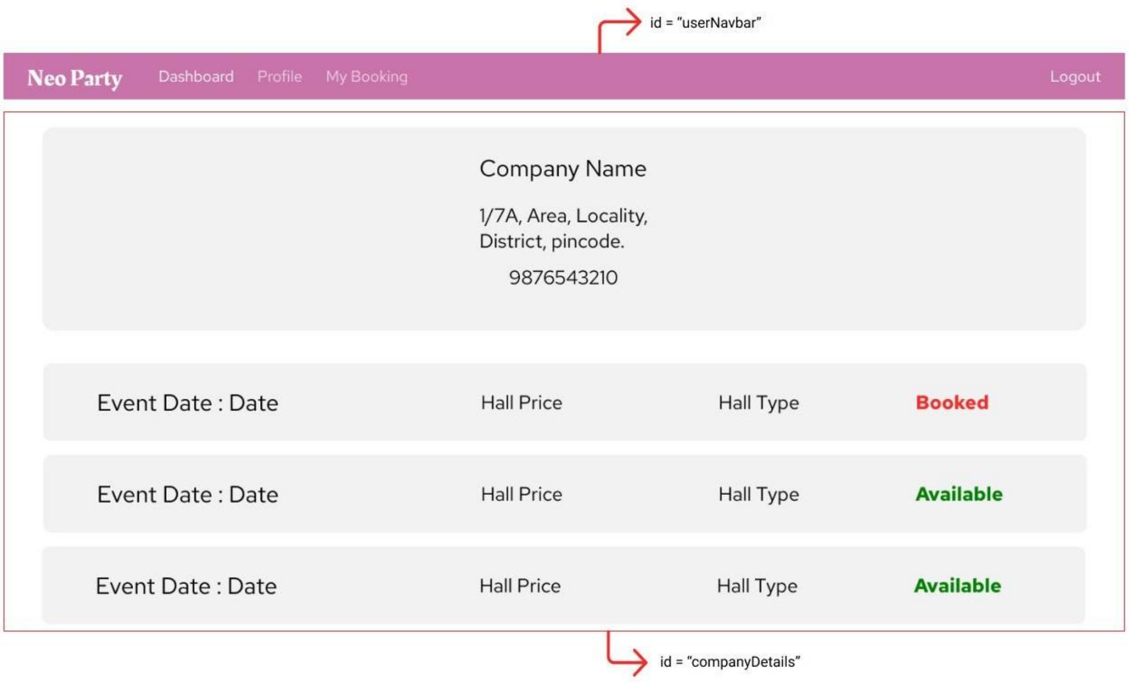
2. Login: Design a login page component where the existing customer can log in using the registered email id and password.
   a. Ids:
      i. loginBox
      ii. userTab
      iii. email
      iv. password
      v. submitButton
      vi. userSignupLink
   b. API endpoint Url: http://localhost:3000/login
   c. Output screenshot:



3. Dashboard / Home: Design a dashboard page component which provides a list of companies available where the user can book a particular Hall.
   a. Ids:
      i. userNavbar
      ii. userDashboard
   b. API endpoint Url: http://localhost:3000/home
   c. Screenshot



4. Company Details : Design a Company Details component which lists available dates along with the individual status of each day (available / Booked).
   a. Ids
      i. userNavbar
      ii. companyDetails
   b. API endpoint URL : [http://localhost:3000/companyDetail/{id]()}
   c. Screenshot



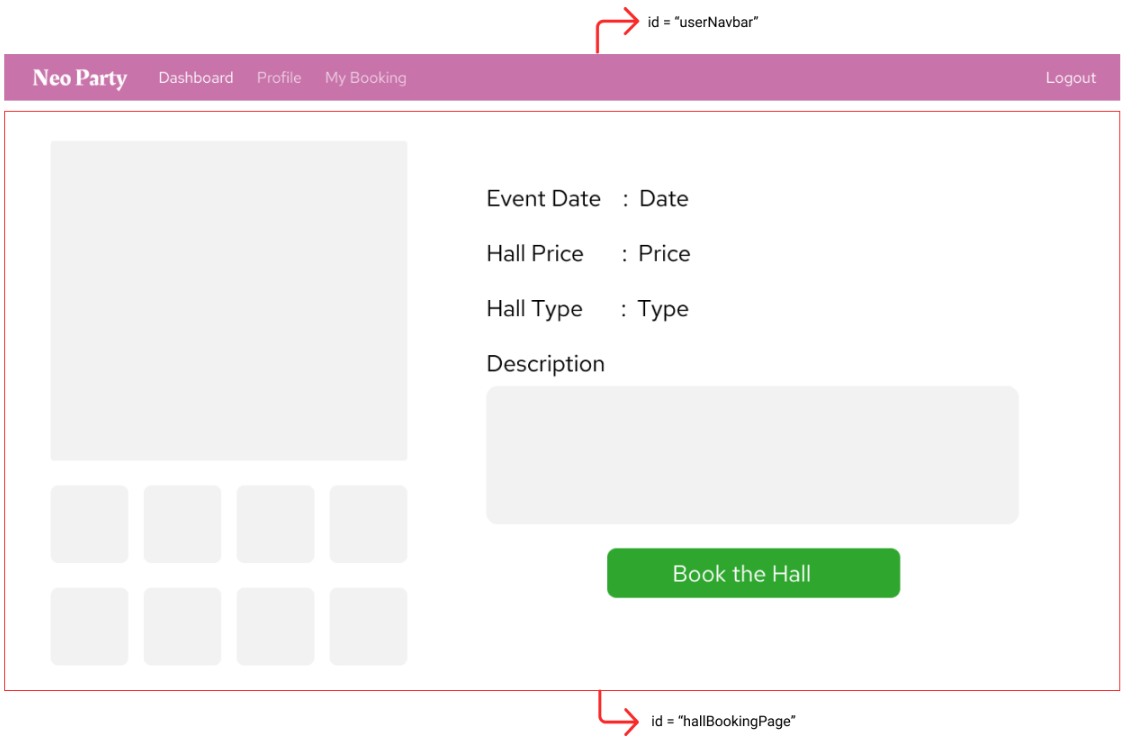
Company Name

1/7A, Area, Locality,
District, pincode.

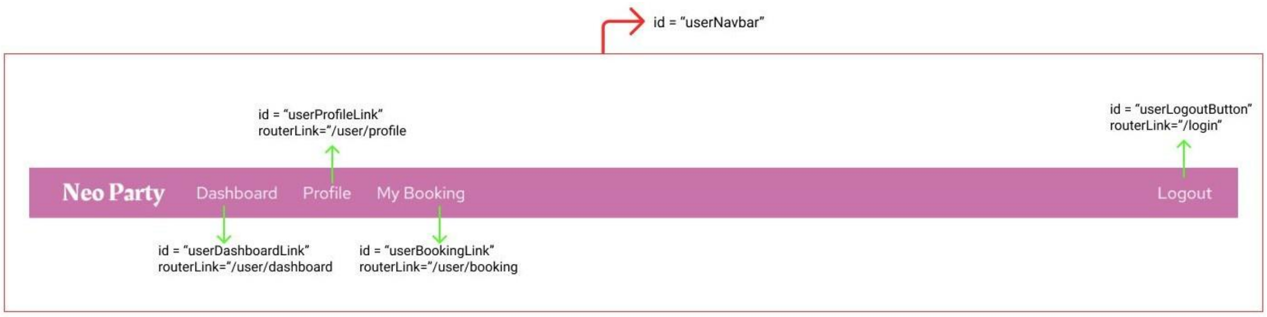
9876543210

Event Date : Date	Hall Price	Hall Type	Booked
Event Date : Date	Hall Price	Hall Type	Available
Event Date : Date	Hall Price	Hall Type	Available

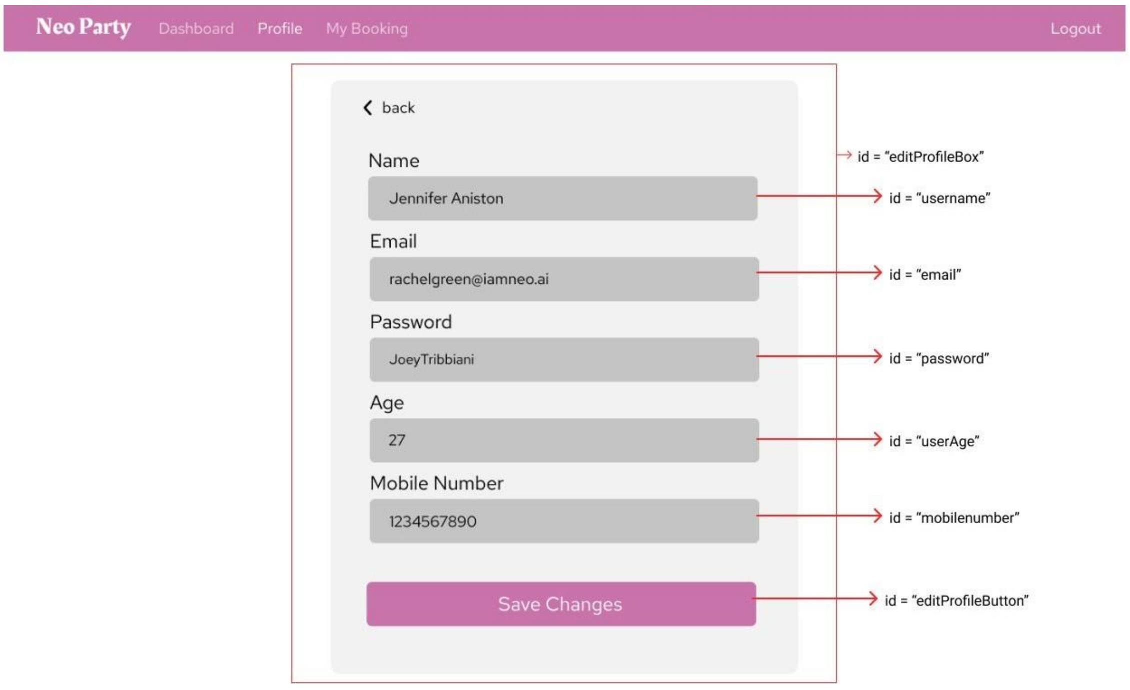
5. Bookings : Design a Booking Page that displays the details of the hall that the user is currently booking.
    a. Ids
        i. userNavbar
        ii. hallBookingPage
    b. API endpoint URL : http://localhost:3000/booking/{id}
    c. Screenshot



6. User Navigation : Design a Booking page component where the user can navigate to their profile and bookings page.
    a. Ids
        i. userNavbar
        ii. userProfleLink
        iii. userDashboardLink
        iv. userBookingLink
        v. userLogoutButton
    b. Screenshot

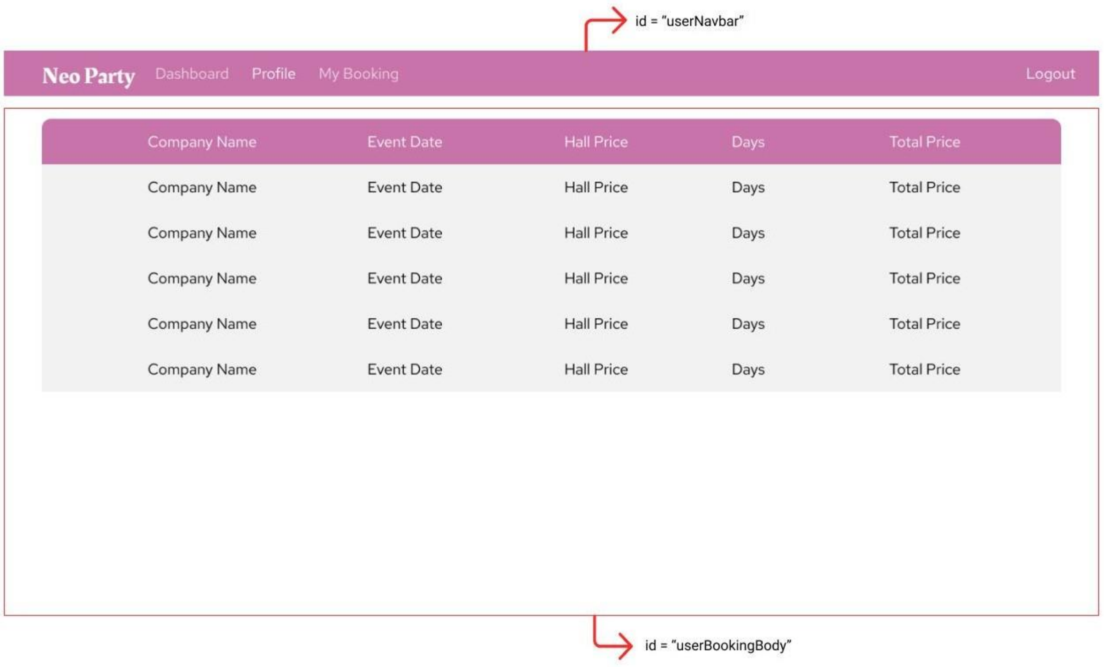


7. User Profile Edit : Design an Profile Edit Page where the user can edit the details of him/herself.
    a. Ids
        i. userNavbar
        ii. editProfileBox
        iii. username
        iv. email
        v. password
        vi. userAge
        vii. mobilenumber
        viii. editProfileButton
    b. API endpoint URL : http://localhost:3000/editProfile/{id}
    c. Screenshot:



Name	Jennifer Aniston
Email	rachelgreen@iamneo.ai
Password	JoeyTribbiani
Age	27
Mobile Number	1234567890

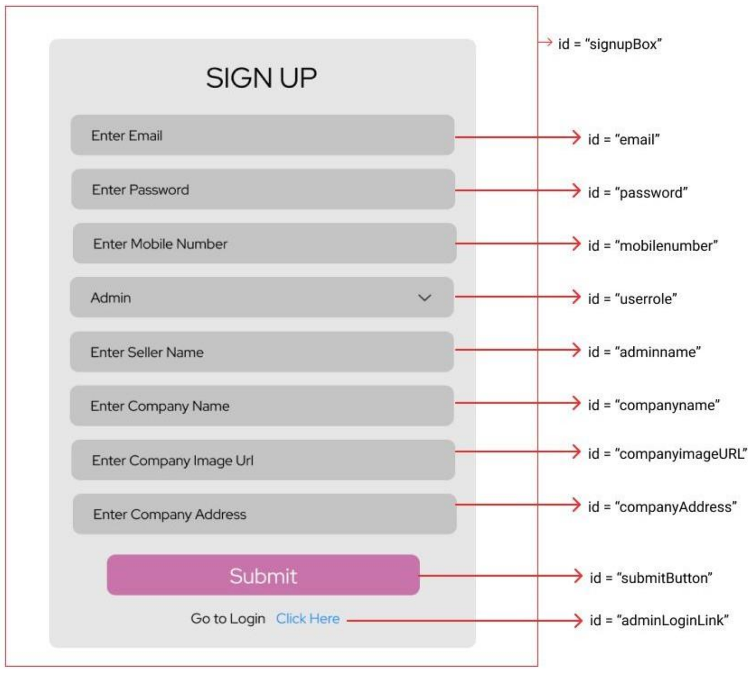
8. User Bookings : Design a User Bookings Page that allows the user to view the bookings that the user has made till date.
    a. Ids:
        i. userNavbar
        ii. userBookingBody
    b. API endpoint URL : http://localhost:3000/bookings/{id}
    c. Screenshot:



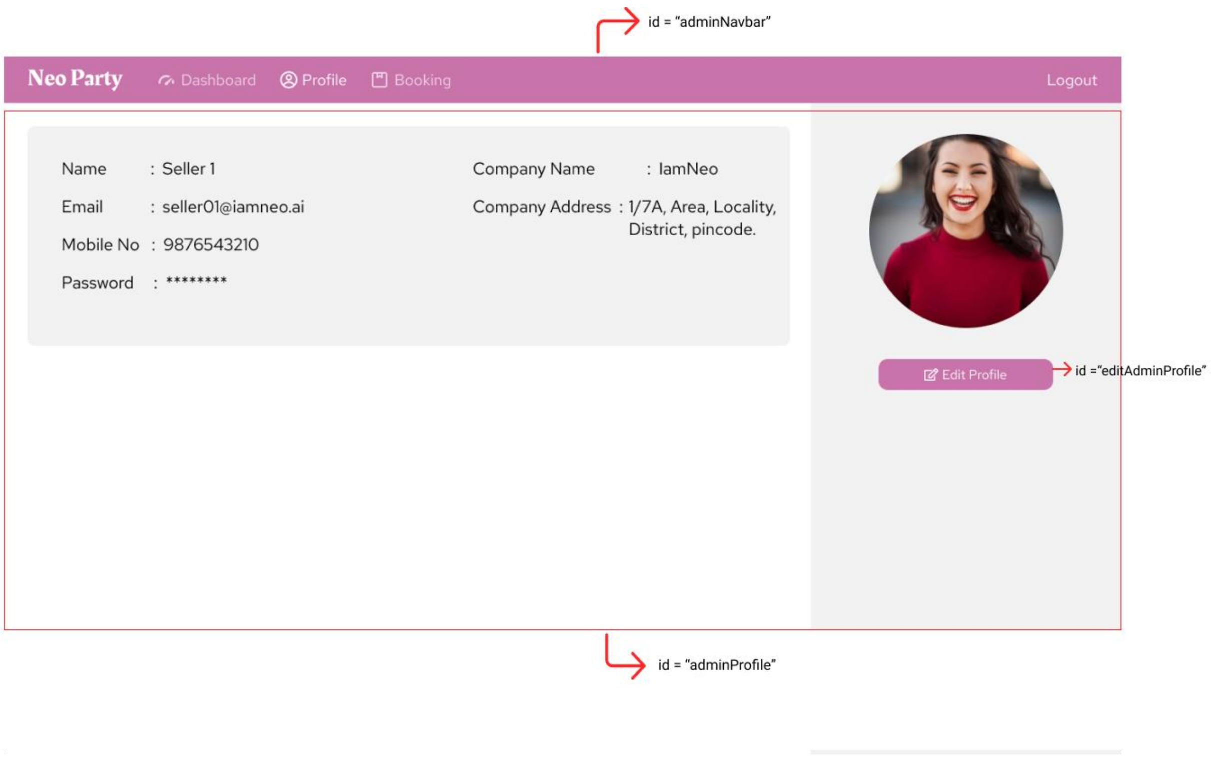
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price

Admin:

9. Admin Signup : Design a Signup page that registers a particular user with the role of Admin.
    a. Ids
        i. signupbox
        ii. email
        iii. password
        iv. mobilenumber
        v. userrole
        vi. adminname
        vii. companyname
        viii. companyimageURL
        ix. companyAddress
        x. submitButton
        xi. adminLoginLink
    b. API endpoint Url : http://localhost:3000/admin/signup
    c. Screenshot

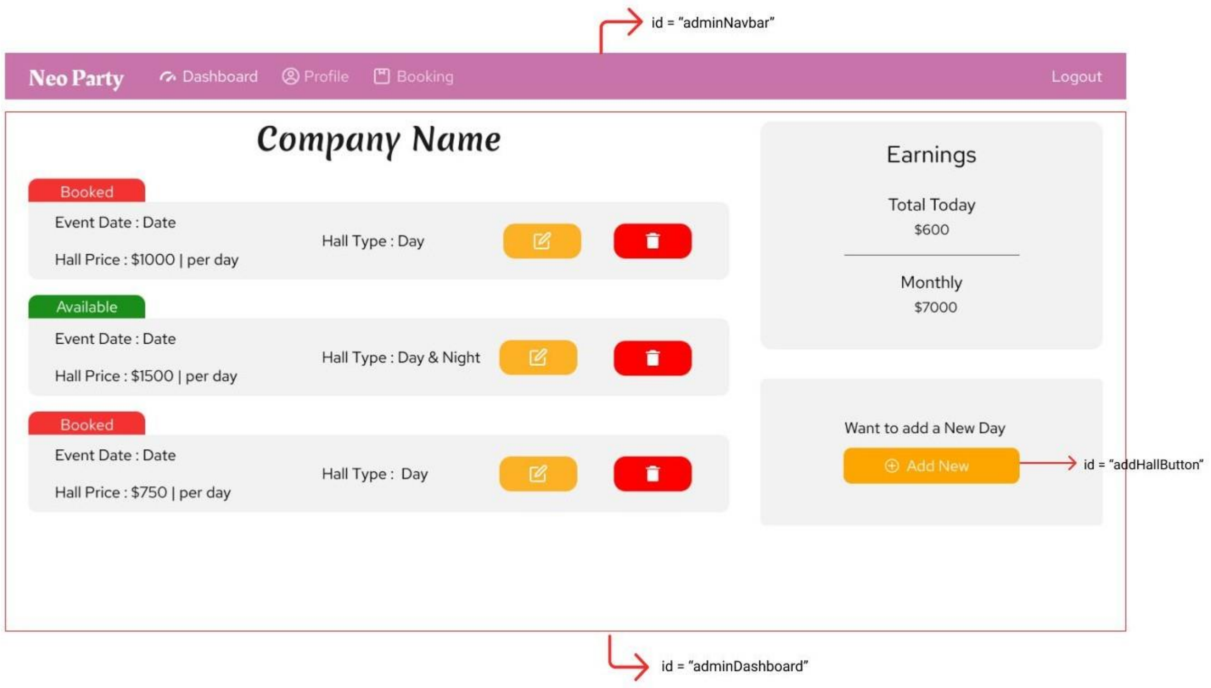


10. Admin Profile: Design a Admin Profile page that displays the details of the admin currently logged in.
    a. Ids
        i. adminNavbar
        ii. adminProfile
        iii. editAdminProfile
    b. API endpoint URL: http://localhost:3000/admin/profile
    c. Screenshots

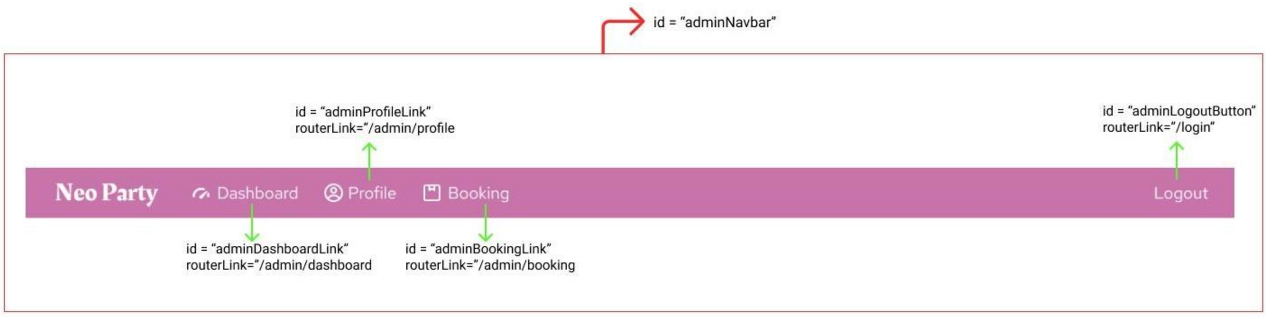


Name : Seller 1
Email : seller01@iamneo.ai
Mobile No : 9876543210
Password : ********
Company Name : IamNeo
Company Address : 1/7A, Area, Locality, District, pincode.

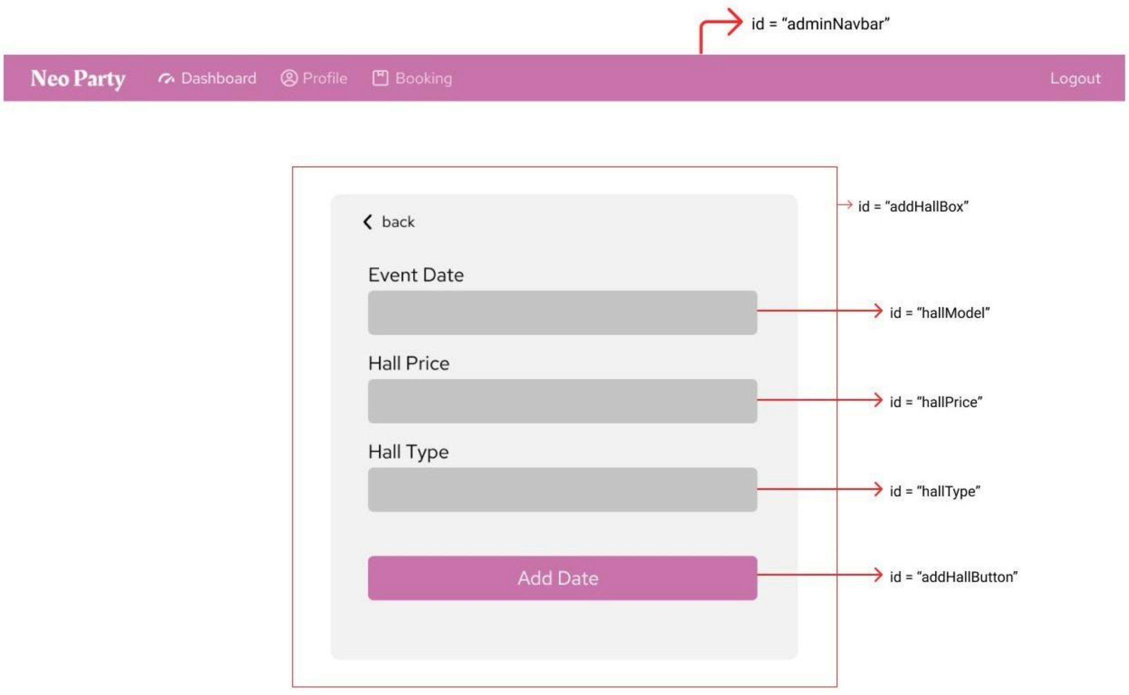
11. Admin Dashboard: Design a dashboard page where the list of products is displayed on the admin side.
    a. Ids
        i. adminNavbar
        ii. addHallButton
        iii. adminDashboard
    b. API endpoint Url: http://localhost:3000/admin/dashboard
    c. Screenshot



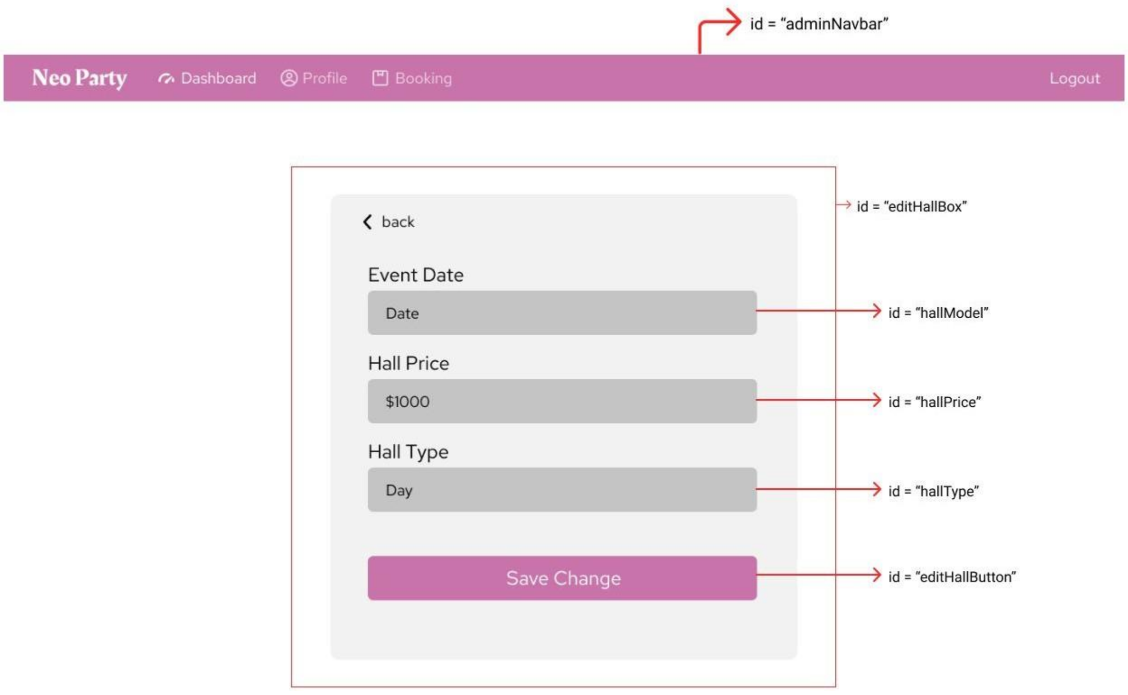
12. Admin Navigation: Design an Admin navigation component that can navigate to the Dashboard, profile and bookings done by the users
    a. Ids:
        i. adminNavbar
        ii. adminProfileLink
        iii. adminDashboardLink
        iv. adminBookingLink
        v. adminLogoutButton
    b. Screenshot:



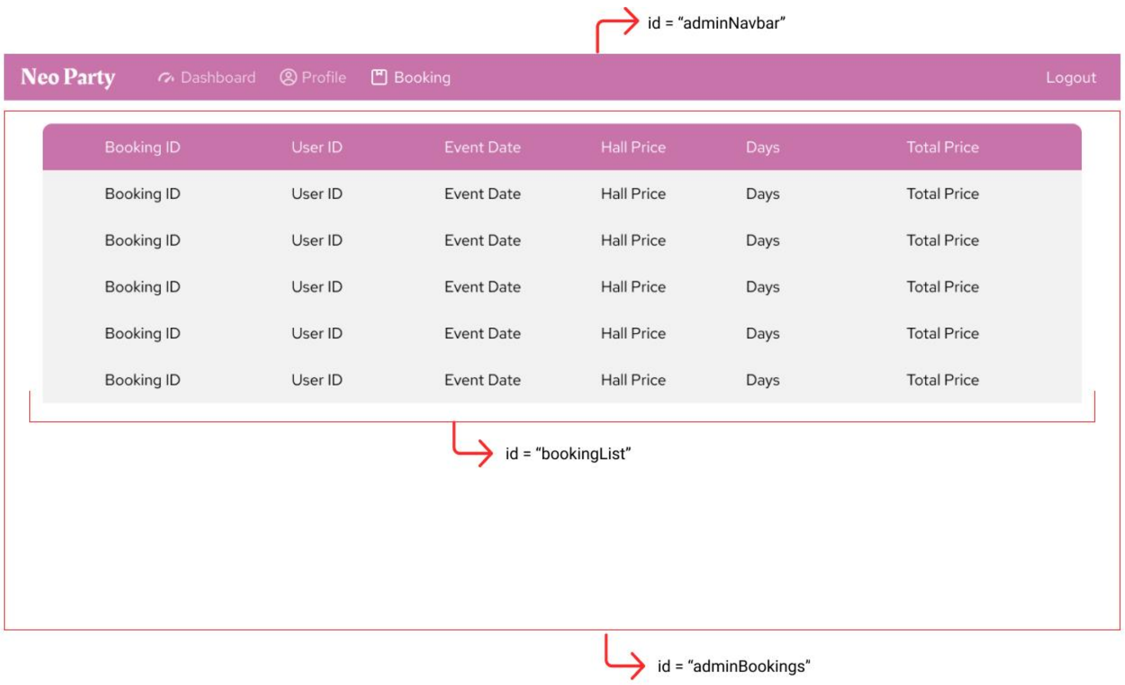
13. Admin Add Date: Design an Add Date component in which the admin can add new dates to the inventory.
    a. Ids:
        i. adminNavbar
        ii. addHallBox
        iii. hallNo
        iv. hallPrice
        v. hallType
        vi. addHallButton
    b. API endpoint Url: http://localhost:3000/admin/addHall
    c. Screenshot



14. Admin Edit Bike: Design an Edit Bike component in which the admin can edit details of an already existing bike in the inventory.
    a. Ids:
        i. adminNavbar
        ii. editHallBox
        iii. hallNo
        iv. hallPrice
        v. hallType
        vi. editHallButton
    b. API endpoint Url: http://localhost:3000/admin/editHall
    c. Screenshot

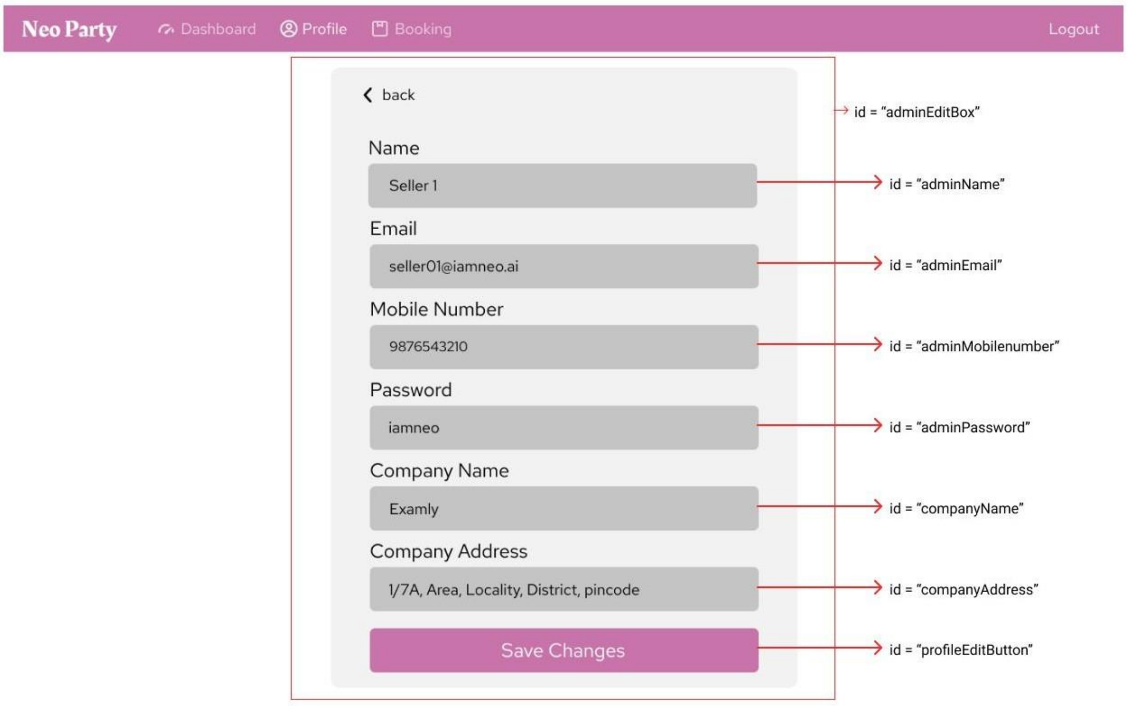


15. Admin Bookings : Design a Admin Bookings component where the admin can see the booking details that all the users have made till present date.
    a. Ids
        i. adminNavbar
        ii. adminBookings
        iii. bookingList
    b. API endpoint URL : http://localhost:3000/admin/bookings
    c. Screenshot:



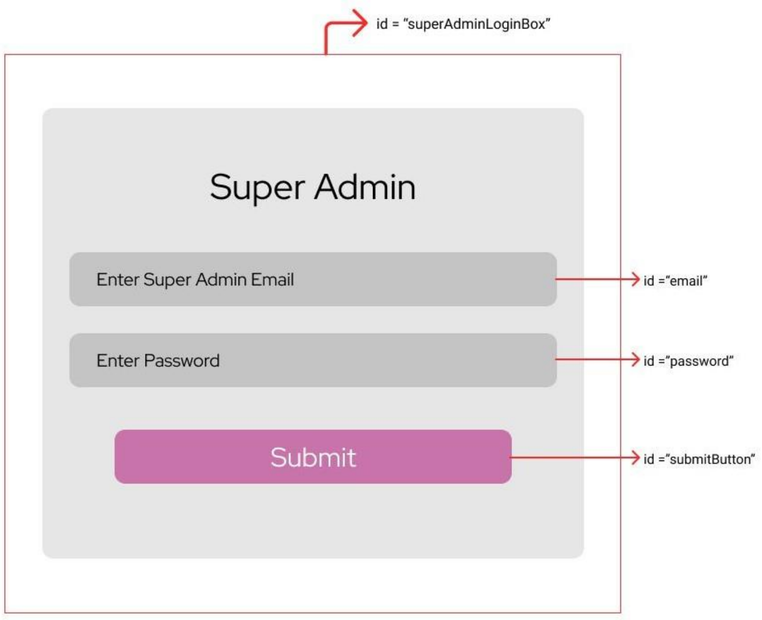
Booking ID	User ID	Event Date	Hall Price	Days	Total Price
Booking ID	User ID	Event Date	Hall Price	Days	Total Price
Booking ID	User ID	Event Date	Hall Price	Days	Total Price
Booking ID	User ID	Event Date	Hall Price	Days	Total Price
Booking ID	User ID	Event Date	Hall Price	Days	Total Price
Booking ID	User ID	Event Date	Hall Price	Days	Total Price

16. Admin Profile Edit : Design an Admin Profile Edit page where the admin can edit the details of himself.
    a. Ids
        i. adminEditBox
        ii. adminName
        iii. adminEmail
        iv. adminMobileNumber
        v. adminPassword
        vi. companyName
        vii. companyAddress
        viii. profileEditButton
    b. API endpoint URL : http://localhost:3000/admin/editProfile
    c. Screenshot:

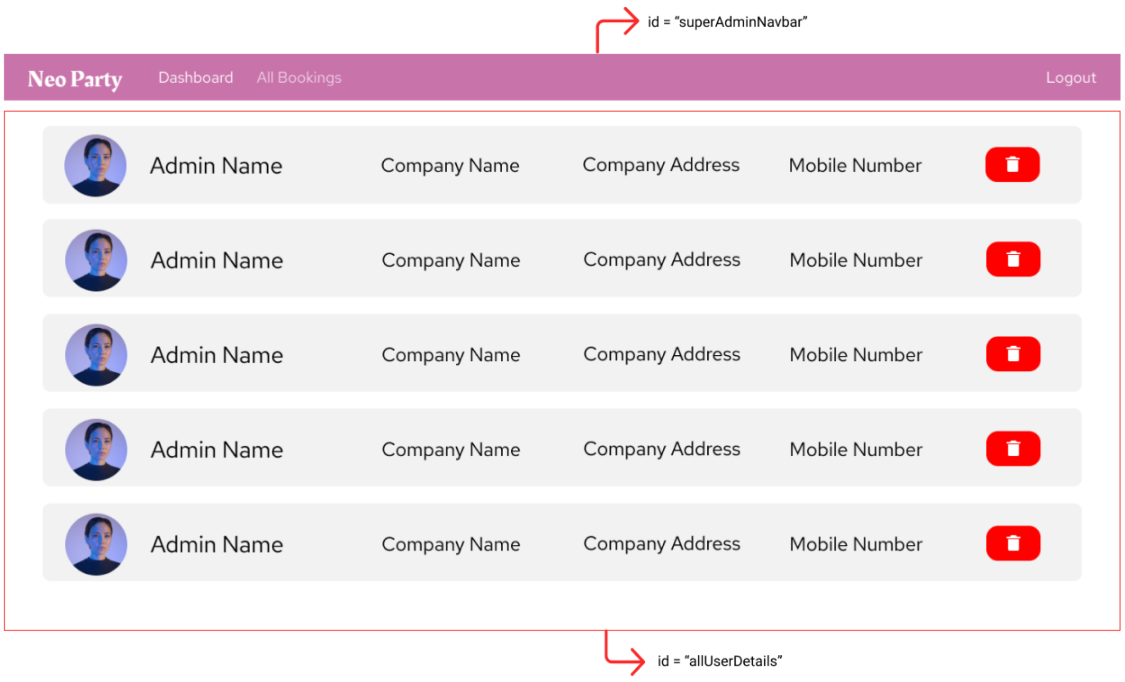


Super Admin:

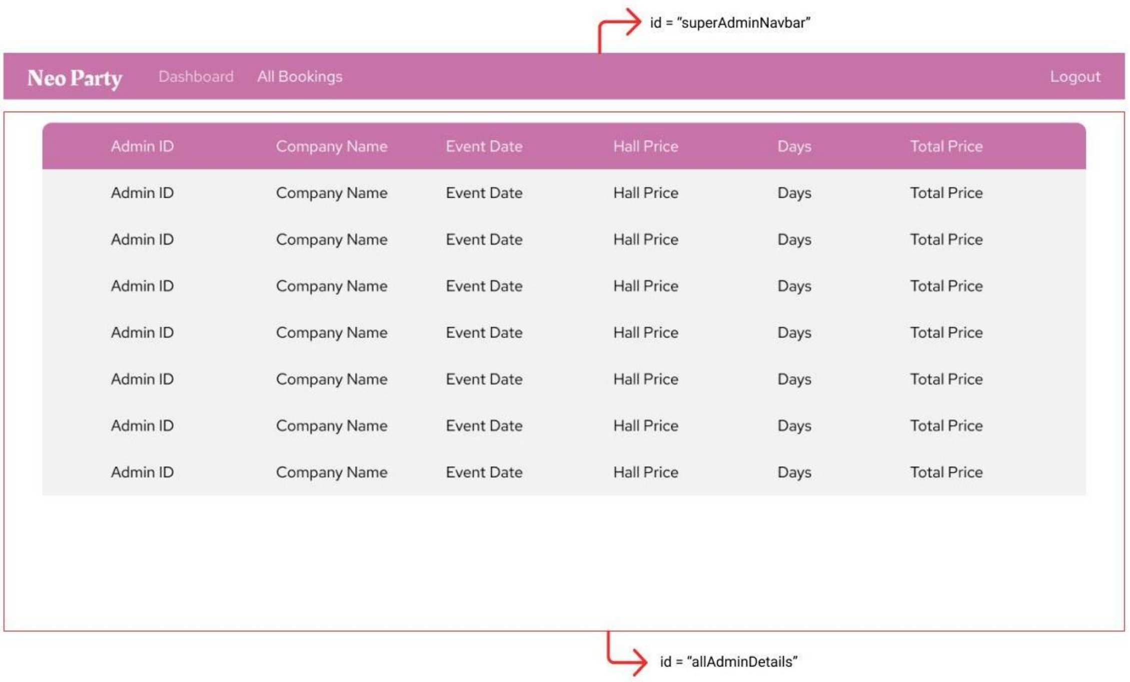
17. Super Admin Login Page: Design a Login page through which the super user can log in.
    a. Ids:
        i. superAdminLoginBox
        ii. email
        iii. password
        iv. submitButton
    b. API endpoint URL : http://localhost:3000/superadmin/login
    c. Screenshot:



18. Super Admin Users page : Design a Super-Admin users page where the super admin has authority to view and delete any user currently registered into the system.
    a. Ids:
        i. superAdminNavbar
        ii. allUserDetails
    b. API endpoint URL : http://localhost:3000/superadmin/users
    c. Screenshot



19. Super Admin Bookings page : Design a Super-Admin Booking component where the super admin has authority to view the bookings.
    a. Ids:
        i. superAdminNavbar
        ii. allAdminDetails
    b. API endpoint URL : http://localhost:3000/superadmin/admin
    c. Screenshot

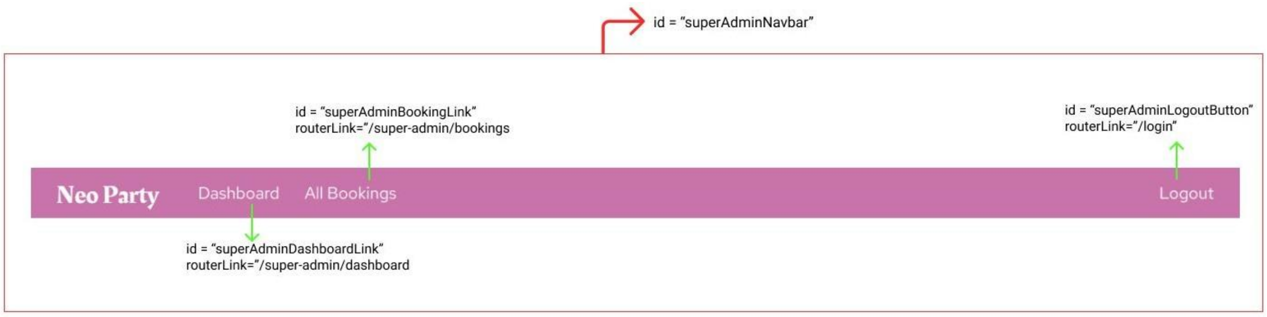

id = "superAdminNavbar"

Neo Party | Dashboard | All Bookings | Logout

Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price

id = "allAdminDetails"

20. Super Admin Navigation : Design a Navigation page for the super user where he can navigate to the dashboard, all the bookings and logout.
    a. Ids :
        i. superAdminNavbar
        ii. superAdminBookingLink
        iii. superAdminLogoutButton
        iv. superAdminDashboardLink
    b. Screenshot:


id = "superAdminNavbar"
id = "superAdminBookingLink"
routerLink="/super-admin/bookings
id = "superAdminLogoutButton"
routerLink="/login"
Neo Party
Dashboard
All Bookings
Logout
id = "superAdminDashboardLink"
routerLink="/super-admin/dashboard

<u>Backend:</u>

<u>Class and Method description:</u>

<u>Model Layer:</u>

1. UserModel: This class stores the user type (admin or the customer) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. username: String
        iv. mobileNumber: String
        v. age: int
        vi. userRole: String
2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String

3. AdminModel: This class stores the details of the product.
    a. Attributes:
        i. email:String
        ii. password:String
        iii. mobileNumber:String
        iv. sellerName:String
        v. userRole:String
        vi. hotelName:String
        vii. hotelImageURL:String
        viii. hotelAddress: String
        ix. earnings:int
4. RoomModel: This class stores the cart items.
    a. Attributes:
        i. roomID:String
        ii. roomNo:String
        iii. adminID:String
        iv. status:String
        v. price:String
        vi. type:String

Controller Layer:

5. AuthController: This class control the user /admin signup and signin
    a. Methods:
        i. isUserPresent(LoginModel data): This method helps to check whether the user present or not and check the email and password are correct and return the boolean value.
        ii. isAdminPresent(LoginModel data): This method helps to check whether the admin present or not and check the email and password are correct and return the boolean value.
        iii. saveUser(UserModel user): This method helps to save the user data in the database.
        iv. saveAdmin(UserModel user): This method helps to save the admin data in the database.
6. RoomController: This class controls the save/edit/delete/view rooms.

a. Methods:

   i. saveRoom(RoomModel data): This method helps the admin to save new rooms in the database.

   ii. editRoom(RoomModel data):This method helps the admin to edit the details of the rooms and save it again in the database.

   iii. deleteRoom(String Room ID): This method helps the admin to delete a room from the hotel and as well as in the database.

   iv. getRooms(String Admin_Email_ID): This method helps the admin to get all the rooms in the hotel.

   v. bookRoom(String Room_ID): This method helps the user to book the room by changing the room status and add the record in the bookings table.

7. AdminController: This class controls the edit/view admin details.

   a. Methods:

      i. editAdmin(AdminModel data): This method helps the admin to edit the profile details and save it in the database.

      ii. getProfile(String Admin_Email_ID): This method helps admin to get the profile details.

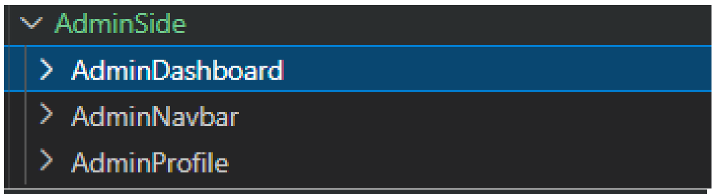
8. UserController: This class helps to get the rooms.

   a. Methods:

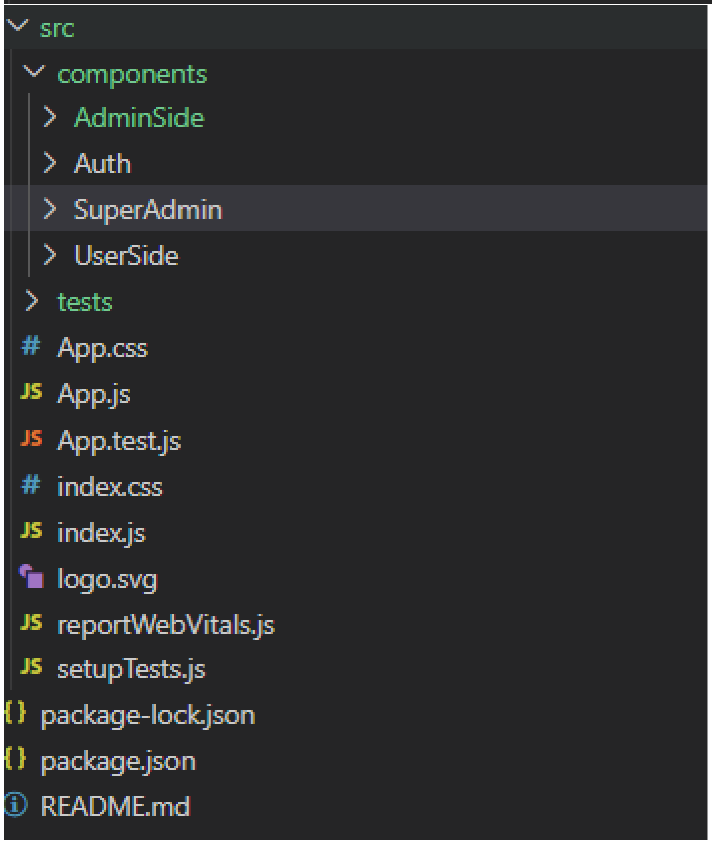
      i. userBookings(String User_ID):This method helps users to get all the booking details done by them.

React Folder Structure:

Admin Side:

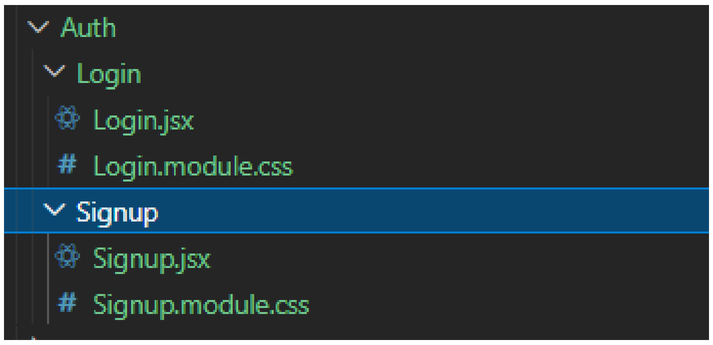


```
AdminSide
  AdminDashboard
  AdminNavbar
  AdminProfile
```



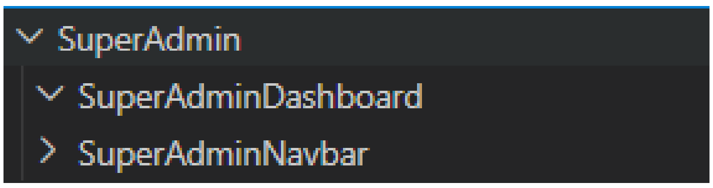
```
src
  components
    AdminSide
    Auth
    SuperAdmin
    UserSide
  tests
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
package-lock.json
package.json
README.md
```

Auth Side:



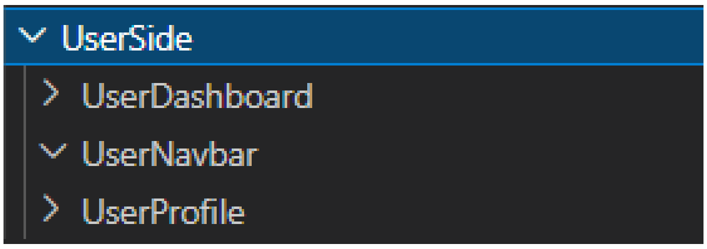
```
Auth
  Login
    Login.jsx
    Login.module.css
  Signup
    Signup.jsx
    Signup.module.css
```

Super Admin Side:



```
SuperAdmin
  SuperAdminDashboard
  SuperAdminNavbar
```

User Side:



```
UserSide
  UserDashboard
  UserNavbar
  UserProfile
```

React component Structure:

- src (folder)
  - components (folder)
    - Navbar (folder)
      - Navbar.jsx (jsx file)
      - Navbar.module.css (css file)

NOTE:

You should create the above folder structure mandatorily to pass the test cases and you can also create extra components if you need.

Workflow Prototypes:

Admin Flow
https://www.figma.com/proto/5QVhQYuw1tYAKFFmtVksCc/Neo-Party-Hall-Admin-Flow?node-id=1%3A482&viewport=556%2C197%2C0.03661709651350975&scaling=scale-down

User Flow
https://www.figma.com/proto/f9BG3VvxHTOTpzXNi9C5zy/Neo-Party-Hall-User-Flow?node-id=1%3A33&viewport=476%2C203%2C0.04019034653902054&scaling=scale-down

Super Admin Flow
https://www.figma.com/proto/B15Jf2uQhqqufVwEgbfrFv/Neo-Party-Hall-Super-Admin-Flow?node-id=1%3A2&viewport=418%2C363%2C0.03856455534696579&scaling=scale-down